REMARKS

The specification has received an objection. Claims 1-9 and 11-20 stand rejected under 35 U.S.C. 102(b). Claim 10 stands rejected under 35 U.S.C. 103. The specification has been objected to. Claims 1-20 have been amended. Claims 21 and 22 have been added.

The specification has received an objection based on page 27, lines 1-2. The specification, page 27, lines 1-2 as amended states that "a finite automaton with output actions is created." Therefore, applicants submit that the objection to the specification is overcome.

Claims 1-9 and 11-20 stand rejected under 35 U.S.C. 102(b) based on "Synthesizing Converters Between Finite State Protocols" by J. Akella and K. McMillan and published in Proceedings of the International Conference on Computer Design, pp. 410-413 (October 14-15, 1991) ("Akella").

The Office action, on page 2, states that "[Akella] discloses two finite state machines and a third finite state machine for the valid data transfer (specification page 5, lines 3-4)." The specification states that Akella discloses:

> Protocols are described as two finite state machines, while a third FSM represents the valid transfer of data. The product machine is taken and pruned of the invalid/useless states. One limitation in this conventional approach is that data width mismatch cannot be handled and that the designer must manually enter the intended behavior of the interface in the form of the third FSM.

(Specification page 5). Clearly, a problem with Akella is that "the designer must manually enter the intended behavior of the interface in the form of the third FSM. Akella discloses this problem:

> In this work the finite state machine part of the protocol converter is obtained similar to the product of the general labeled transition systems (LTSs). . . . a third LTS C which represents the desired

sequence of inter-operation between the interfaces is specified. . . .

(Page 411, 2nd column). Clearly, in Akella, "a third LTS C which represents the desired sequence of inter-operation between the interfaces is specified" by the designer, which is the problem in Akella. Because Akella requires a "desired sequence of inter-operation between the interfaces to be specified," Akella does not disclose "automatically synthesizing an interface between the first and second protocols based on the first and second finite automatons."

In contrast, claim 1 as amended recites "automatically synthesizing an interface between the first and second protocols based on the first and second finite automatons." Therefore, applicants submit that claim 1 as amended is patentable over Akella.

Given that claims 2-9, 11, and 21 depend from claim 1 as amended, applicants submit that these claims are also patentable over Akella.

Claim 12 stands rejected under 35 U.S.C. 102(b) based on Akella.

Akella does not disclose "a synthesizing unit to automatically synthesize an interface between the first and second protocols based on the first and second finite automatons," as recited in claim 12 as amended. Therefore, applicants submit that claim 12 as amended is patentable over Akella.

Given that claims 13-19 depend from claim 12 as amended, applicants submit that these claims are also patentable over Akella.

Claim 20 stands rejected under 35 U.S.C. 102(b) based on Akella.

Akella does not disclose "automatically synthesizing an interface between the first and second protocols based on the first and second finite automatons," as recited in claim 20 as amended. Therefore, applicants submit that claim 20 as amended is patentable over Akella.

Claim 10 stands rejected under 35 U.S.C. 103 based on Akella in view of Official notice that it "is commonly well-known practice to one in the computer art to incorporate the priority into any computer design." (page 7 of the Office action).

Akella does not disclose "automatically synthesizing an interface between the first and second protocols based on the first and second finite automatons," as recited in claim 1 as amended.

Official notice does not disclose "automatically synthesizing an interface between the first and second protocols based on the first and second finite automatons," as recited in claim 1 as amended.

Even if Akella and Official notice were combined, the combination would neither teach nor suggest "automatically synthesizing an interface between the first and second protocols based on the first and second finite automatons," as recited in claim 1 as amended.

Therefore, applicants submit that claim 1 as amended is patentable over Akella in view of Official notice. Given that claim 10 depends from claim 1 as amended, applicants submit that this claim is also patentable over Akella in view of Official notice.

CONCLUSION

On the basis of the above remarks, reconsideration and allowance of the claims is believed to be warranted and such action is respectfully requested. If the Examiner has any questions or comments, the Examiner is invited to contact the undersigned at the number listed below.

DATE: February 13, 2003

Respectfully submitted,

By: _____
Jeffrey S. Smith
Registration No. 39,377

Bingham McCutchen LLP
Three Embarcadero Center, Suite 1800
San Francisco, California 94111
Telephone: (650) 849-4422
Telefax: (650) 849-4800

Express Mail Label No.
EV 154 656 530 US

Enclosures: Marked up version of amended claims pursuant to 37 C.F.R. § 121(c)(1)(ii).

Attachment A

Version With Markings To Show Changes

In the Specification

Please replace the paragraph starting on page 26, line 16 with the following rewritten paragraph.

After generating 204 the finite automata, the interface generator 118 of the present invention determines 206 the permitted sequence of operations of the two protocols and resolves 208 all non-determinisms. The two generated 204 deterministic finite automata recognize a transaction on either side of the interface. The present invention then proceeds with the construction 206 of a subset of the product machine that performs the correct transfer of data. Moreover, the input/output nature of the signals is taken into account, so that a finite [state machine rather than a finite automation] automation with output actions is created.

Attachment B

Version With Markings To Show Changes

In the Claims

1. (Amended) A method for exchanging data messages between a first block having a first protocol for exchanging messages and a second block having a second protocol for exchanging messages, the method comprising: [the steps of]

receiving a first representation, representing the first protocol, said first representation using regular expressions;

receiving a second representation, representing the second protocol, said second representation using regular expressions;

generating a first finite automaton for said first representation;

generating a second finite automaton for said second representation; and

automatically synthesizing an interface between the first and second protocols based on the first and second finite automatons.

[generating a third representation, representing one or more permitted operations of said first and second finite automata; and

eliminating any non-determinisms in said third representation to generate an interface between said first and second protocols.

2. (Amended) The method of claim 1, further comprising: [the step of:]

automatically corresponding data from said first and second protocols.

3. (Amended) The method of claim 2, further comprising: [the step of:]

automatically translating data between said first protocol and said second protocol, said data in said first protocol having a first sequence, said data in said second protocol having a second sequence that is different from said first sequence.

4. (Amended) The method of claim 2, wherein said [step of] generating a first finite automaton [includes the steps of] comprises:

identifying the initial state of the first protocol;

identifying a first sequence of data according to the first protocol;

constructing derivatives of [said] regular expressions; and

eliminating equivalent expressions.

5. (Amended) The method of claim 4, wherein said [step of] identifying a first sequence of data comprises: [includes the steps of:]

collecting data that is transferred during one or more transitions; and

integrating said data with previous transitions.

6. (Amended) The method of claim 5, further comprising: [the step of:]

automatically translating data between said first protocol to said second protocol, said data in said first protocol having a first sequence, said data in said second protocol having a second sequence that is different from said first sequence.

7. (Amended) The method of claim 1, further comprising: [the step of:]

automatically translating data between said first protocol to said second protocol, said data in said first protocol having a first sequence, said data in said second protocol having a second sequence that is different from said first sequence.

8. (Amended) The method of claim 21, [1,] wherein [the step of] automatically generating a third representation comprises: [includes the steps of:]

(a) selecting an interface state representing a first finite automaton state and a second finite automaton state;

(b) identifying all outgoing transitions in said selected state;

(c) determining a new state for each outgoing transition;

(d) repeating steps (a)-(c) for each interface state.

9. (Amended) The method of claim 8, wherein [the step of] generating a third representation comprises: [includes the steps of:]

identifying said permitted operations as operations that do not result in a data inconsistency.

10. (Amended) The method of claim 8, wherein said eliminating comprises: [step includes the steps of:]

identifying non-deterministic transitions for each interface state;

selecting a single outgoing transition for each interface state for each input value

based upon priority parameters to generate a deterministic interface between the first and second protocols.

11. (Amended) The method of claim 1, wherein said [step of] generating a first finite automaton comprises: [includes the steps of:]

identifying the initial state of the first protocol;

identifying a first sequence of data according to the first protocol;

constructing derivatives of [said] regular expressions; and

eliminating equivalent expressions.

12. (Amended) A computer based system for exchanging data messages between a first block having a first protocol for exchanging messages and a second block having a second protocol for exchanging messages, the system [method] comprising: [the steps of]

a storage device[, for storing data, and sequences of operations] to store data and sequences of operations;

a processor[, disposed] to receive signals from said storage device[, for executing] and to execute said sequences of operations;

a receiving unit[, disposed] to transmit signals to said processor[, for receiving] and to receive a first and second representation[, representing] of the first and second protocols;[, said first and second representations using regular expressions;]

an automata unit[, for generating] to generate a first finite automaton for said first representation and [for generating] to generate a second finite automaton for said second

representation; and

a synthesizing unit to automatically synthesize an interface between the first and second protocols based on the first and second finite automatons.

[a product unit, for generating a third representation, representing one or more permitted operations of said first and second finite automata; and

a determinism unit, for eliminating any non-determinisms in said third representation.]

13. (Amended) The system of claim 12, further comprising:

a corresponding unit[, disposed] to receive signals from said processor[, for] and to automatically correspond[ing] data from said first and said second protocol.

14. (Amended) The system of claim 13, further comprising:

a translation unit[, for] to automatically translate[ing] data between said first protocol and said second protocol, said data in said first protocol having a first sequence, said data in said second protocol having a second sequence that is different from said first sequence.

15. (Amended) The system of claim 13, wherein said automata unit comprises: [includes:]

a first identifying unit [for identifying] to identify the initial state of the first protocol;

a second identifying unit [for identifying] to identify a first sequence of data

according to the first protocol;

a derivative unit [for constructing] to construct derivatives of [said] regular expressions; and

an eliminating unit [for eliminating] to eliminate equivalent expressions.

16. (Amended) The system of claim 15, wherein said second identifying unit comprises: [includes:]

a data collection unit [for collecting] to collect data that is transferred as one or more transitions; and

a data analyzer [for integrating] to integrate said data with previous transitions.

17. (Amended) The system of claim 12, further comprising:

a translation unit[, for automatically translating] to automatically translate data between said first protocol and said second protocol, said data in said first protocol having a first sequence, said data in said second protocol having a second sequence that is different from said first sequence.

18. (Amended) The system of claim 12, wherein the product unit comprises: [includes:]

a selection unit [for selecting] to select an interface state representing a first finite automaton state and a second finite automaton state;

an identifying unit [for identifying all] to identify outgoing transitions in said selected state;

a state unit [for determining] to determine a new state for each outgoing transition;

19. (Amended) The system of claim 18, [claim 8,] wherein the product unit further comprises: [includes:]

a consistency unit [for identifying] to identify said permitted operations as operations that do not result in a data inconsistency.

20. (Amended) A computer readable medium storing instructions which, when executed by a processing system, cause the system to perform a method for exchanging data messages between a first block having a first protocol for exchanging messages and a second block having a second protocol for exchanging messages, the method comprising: [program embodied in a tangible medium and capable of being read by a computer, for performing the method of claim 1.]

receiving a first representation of the first protocol;

receiving a second representation of the second protocol;

generating a first finite automaton for said first representation;

generating a second finite automaton for said second representation;

automatically synthesizing an interface between the first and second protocols based on the first and second finite automatons.

21. (New) The method of claim 1, wherein automatically synthesizing an interface between

the first and second protocols based on the first and second finite automatons further comprises:

automatically generating a third representation, representing one or more permitted operations of said first and second finite automata; and

automatically eliminating any non-determinisms in said third representation to generate an interface between said first and second protocols.

22. (New) a method for exchanging data messages between a first block having a first protocol and a second block having a second protocol, the method comprising:

generating a first finite automaton corresponding to the first protocol;

generating a second finite automaton corresponding to the second protocol;

automatically generating a representation of one or more permitted operations of the first and second finite automaton.